LA-UR- 10-05696

Approved for public release; distribution is unlimited.

Title:	Spacial and Objective Decompositions for Very Large SCAPs
Author(s):	Russell Bent Pascal Van Hentenryck Carleton Coffrin
Intended for:	2011 Informs Computing Society Conference

Los Alamos
NATIONAL LABORATORY
EST. 1943

Form 836 (7/06)

Spacial and Objective Decompositions for Very Large SCAPs

Pascal Van Hentenryck[1], Russell Bent[2], and Carleton Coffrin[1]

[1] Brown University, Providence RI 02912, USA
[2] Los Alamos National Laboratory, Los Alamos NM 87545, USA

Abstract. This paper considers the single commodity allocation problem (SCAP) for disaster recovery, a fundamental problem faced by all populated areas [11]. SCAPs are complex stochastic optimization problems that combine resource allocation, warehouse routing, and parallel fleet routing. Moreover, these problems must be solved under tight runtime constraints to be practical in real-world disaster situations. This paper revisits the SCAP algorithm proposed in [11] and proposes new storage allocation models that are necessary to enable the algorithm to scale to problem sizes of three orders of magnitude greater (250, 500, 1000 storage locations). The new algorithms are validated on large-scale hurricane disaster scenarios generated by Los Alamos National Laboratory using state-of-the-art disaster simulation tools.

1 Background & Motivation

Every year seasonal hurricanes threaten coastal areas. The severity of hurricane damage varies from year to year, but considerable human and monetary resources are always spent to prepare for and recover from these disasters. It is policy makers who make the critical decisions relating to how money and resources are allocated for preparation and recovery. At this time, preparation and recovery plans developed by policy makers are often ad hoc and rely on available subject matter expertise. Furthermore, the National Hurricane Center (NHC) of the National Weather Service in the United States (among others) is highly skilled at generating ensembles of possible hurricane tracks but current preparation methods often ignore this information.

Previous work has been successful in solving this problem more rigorously by combining optimization techniques and disaster-specific information given by NHC predictions [11]. The problem is not only hard from a combinatorial optimization standpoint, but it is also inherently stochastic because the exact outcome of the disaster is unknown. Although humans have difficulty reasoning over uncertain data, recent work in the optimization community [13, 4, 11] has shown that stochastic optimization techniques can overcome this difficulty and find robust solutions to problems with uncertainty. However, prior work on SCAPs only considered problems with up to 100 storage locations, although large-scale planning may require as many as 1000 storage locations. The start-of-the-art algorithm from [11] has difficulties to scaling problems with 250 storage locations and really thrashes for larger instances.

The purpose of this paper is to investigate how to enhance the state-of-the-art algorithms to tackle the large-scale SCAP instances. It demonstrates that spatial and objective decompositions allow the original SCAP algorithm to scale to much larger instances. More precisely, this paper makes the following technical contributions:

1. It investigates scalability issues of the original multi-stage SCAP algorithm.
2. It proposes two enhancements to the multi-stage SCAP algorithm: an aggregate storage model and a sequential storage model. The former model is particularly appropriate for large instances, while the latter is best suited for very large instances.
3. It validates the resulting models on the delivery of potable water for hurricane recovery.

Section 2 of this paper reviews similar work on disaster preparation and recovery problems. Section 3 presents a mathematical formulation of the disaster recovery problem and sets up the notations for the rest of paper. Section 4 reviews the overall approach for solving SCAPs as presented in [11]. Section 5 presents the new stochastic storage formulations using (hopefully) intuitive models. Section 6 reports the experimental results of our algorithm modifications on benchmark instances to validate the approach and Section 7 concludes the paper.

2 Previous Work

The operations research community has been investigating the field of humanitarian logistics since the 1990s but recent disasters have brought increased attention to these kinds of logistical problems [16, 3, 8, 7]. Humanitarian logistics is filled with a wide variety of optimization problems that combine aspects from classic problems in inventory routing, supply chain management, warehouse location, and vehicle routing. The problems posed by humanitarian logistics add significant complexity to their classical variants. The operations research community recognizes that novel research in this area is required to solve these kinds of problems [16, 3]. Some of the key features that characterize these problems are as follows:

1. **Multi-Objective Functions** - High-stake disaster situations often have to balance conflicting objective goals (e.g. operational costs, speed of service, and unserved customers) [2, 6, 1, 10, 11].
2. **Non-Standard Objective Functions** - A makespan time objective in VRPs [2, 5, 11] or equitability objectives [1].
3. **Arbitrary Side Constraints** - Limited resources, a fixed vehicle fleet [1, 11], fixed latest delivery time [2, 1], or a insufficient preparation budget [6, 9, 11].
4. **Stochastic Aspects** - Disasters are inherently unpredictable. Preparations and recovery plans must be robust with respect to many scenarios [6, 10, 11].

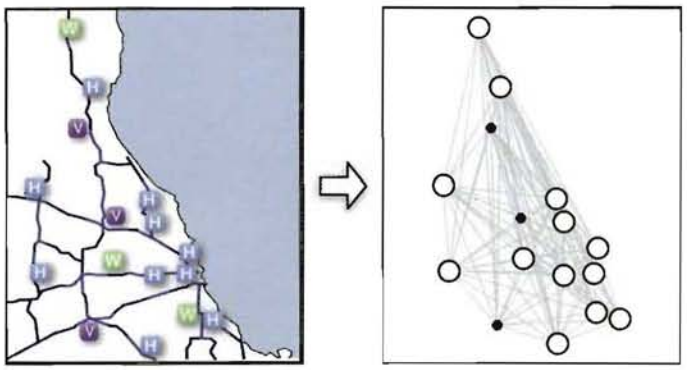

Fig. 1. Populated Area Abstraction

Humanitarian logistics also studies these problems at a variety of scales in both space and time. Some problems consider a global scale with time measured in days and weeks [6], while others focus on the minute-by-minute details of delivering supplies from local warehouses directly to the survivors [2, 1, 11]. This paper considers a scale which is often called the "last mile" of distribution. This involves warehouse selection and customer delivery at the city and state scale.

The operations research community has mainly formulated these problems using MIP models. Many of the humanitarian logistics problems are complex and MIP formulations do not always scale to real world instances [1, 2]. Additionally, it was shown that MIP solvers can have difficulty with some of the unique features of these kinds of problems even when problem sizes are small (e.g., with minimizing the latest delivery time in VRPs [5]). Local search techniques are often used to scale the problems to real world instances [2, 5]. [11] demonstrated that hybrid optimization and decomposition methods can yield high-quality solutions to such challenges and scale to real-world instances. This work extends the work of [11] and shows that additional decomposition can provide significant scaleability. To the best of our knowledge, this is the first time that SCAPs with over 100 storage locations have been solved.

3 The Single Commodity Allocation Problem (SCAP)

In formalizing SCAPs, a populated area is represented as a graph $G = \langle V, E \rangle$ where V represents those sites of interest to the allocation problem, i.e., sites requiring the commodity after the disaster (e.g., hospitals, shelters, and public buildings) and vehicle storage depots. The required commodity can be stored at any node of the graph subject to some side constraints and the graph edges have weights representing travel times. Figure 1 illustrates this transformation. The weights on the edges form a metric space but it is not Euclidean due to the transportation infrastructure. Moreover, the travel times can vary in different disaster scenarios due to road damage. The primary outputs of a SCAP are (1)

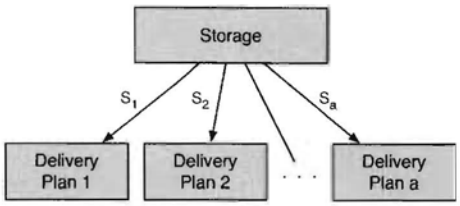

Fig. 2. SCAP Solution

Given:

Repositories: $R_{i \in 1..n}$
- Capacity: RC_i
- Investment Cost: RI_i
- Maintenance Cost: RM_i

Vehicles: $V_{i \in 1..m}$
- Capacity: VC
- Start Depot: D_i^+
- End Depot: D_i^-

Scenario Data: $S_{i \in 1..a}$
- Scenario Probability: P_i
- Available Sites: $AR_i \subset \{1..n\}$
- Site Demand: $C_{i,1..n}$
- Travel Time Matrix: $T_{i,1..l,1..l}$

Weights: W_x, W_y, W_z

Budget: B

Output:

The amount stored at each warehouse
Delivery schedules for each vehicle

Minimize:

$W_x *$ Unserved Demands +
$W_y * MAX^i_{1..m}$Tour Time$_i$+
$W_z *$ Investment Cost +
$W_z *$ Maintenance Cost

Subject To:

Vehicle and site capacities
Vehicles start and end locations
Costs $\leq B$

Notes:

Every warehouse that stores a unit must be visited at least once

Fig. 3. Single Commodity Allocation Problem Specification

the amount of commodity to be stored at each node; (2) for each scenario and each vehicle, the best plan to deliver the commodities (Figure 2 depicts a SCAP solution). Figure 3 summarizes the entire problem, which we now describe in detail.

Objectives The objective function aims at minimizing three factors: (1) The amount of unsatisfied demands; (2) the time it takes to meet those demands; (3) the cost of storing the commodity. Since these values are not expressed in the same units, it is not always clear how to combine them into a single objective function. Furthermore, their relative importance is typically decided by policy makers on a case-by-case basis. For these reasons, this specification uses weights W_x, W_y, and W_z to balance the objectives and to give control to policy makers.

Side Constraints The first set of side constraints concerns the nodes of the graph which represent the repositories in the populated area. Each repository $R_{i \in 1..n}$ has a maximum capacity RC_i to store the commodity. It also has a one-time

initial cost RI_i (the investment cost) and an incremental cost RM_i for each unit of commodity to be stored. As policy makers often work within budget constraints, the sum of all costs in the system must be less than a budget B.

The second set of side constraints concerns the deliveries. We are given a fleet of m vehicles $V_{i \in 1..m}$ which are homogeneous in terms of their capacity VC. Each vehicle has a unique starting depot D_i^+ and ending depot D_i^-. Unlike classic vehicle routing problems [15], customer demands in SCAPs often exceed the vehicle capacity and hence multiple deliveries are often required to serve a single customer.

Stochasticity SCAPs are specified by a set of a different disaster scenarios $S_{i \in 1..a}$, each with an associated probability P_i. After a disaster, some sites are damaged and each scenario has a set AR_i of available sites where the stored commodities remain intact. Moreover, each scenario specifies, for each repository R_i, the demand C_i. Note that a repository may have a demand even if that repository is not available. Finally, site-to-site travel times $T_{i,1..l,1..l}$ (where $l = |V|$) are given for each scenario and capture infrastructure damages.

Unique Features Although different aspects of this problem were studied before in the context of vehicle routing, location routing, inventory management, and humanitarian logistics, SCAPs present unique features. Earlier work in location-routing problems (LRP) assumes that (1) customers and warehouses (storage locations) are disjoint sets; (2) the number of warehouses is $\approx 3..10$; (3) customer demands are less than the vehicle capacity; (4) customer demands are atomic

None of these assumptions hold in the SCAP context. In a SCAP, it may not only be necessary to serve a customer with multiple trips but, due to the storage capacity constraints, those trips may need to come from different warehouses. The key features of SCAP are: (1) each site can be a warehouse and/or customer; (2) one warehouse may have to make many trips to a single customer; (3) one customer may be served by many warehouses; (4) the number of available vehicles is fixed; (5) vehicles start and end in different depots; (6) the objective is to minimize the time of the last delivery. Minimizing the time of the last delivery is a very difficult aspect of this problem as in demonstrated in [5], but we will show that the stochastic storage decisions quickly become the most difficult aspect as the number of storage locations increases.

4 The Basic Approach

This section reviews the state-of-the-art algorithm for solving the SCAP problem, which is discussed in detail in [11]. Our extensions to this work are covered in Section 5. Due to the complex stochastic nature of the SCAP problem the previous work proposed a multi-stage algorithm that decomposes the storage, customer allocation, and routing decisions. The stages and the key decisions of each stage are as follows:

```
MULTI-STAGE-SCAP(G)
1  D ← StochasticStorageProblem(G)
2  for s ∈ 1..a
3  do C ← CustomerAllocationProblem(G_s, D_s)
4     for w ∈ 1..n
5     do T ← RepositoryPathRoutingProblem(G_s, C_w)
6     I ← AggregateFleetRouting(G_s, T)
7     S_s ← PathBasedFeetRouting(G_s, T, I)
8  return S
```

Fig. 4. The Hybrid Stochastic Optimization Algorithm for SCAPs.

1. **Storage**: Which repositories store the commodity and how much do they store?
2. **Customer Allocation**: How is the stored commodity allocated to each customer?
3. **Repository Routing**: For each repository, what is the best customer distribution plan?
4. **Fleet Routing**: How to visit the repositories to minimize the time of the last delivery?

The decisions of each stage are considered independently and use the optimization technique most appropriate to their nature. The first two stages are formulated as MIPs, the third stage is solved optimally using constraint programming, and the forth stage uses large neighborhood search (LNS). The algorithm presented in [11] for solving a SCAP instance $\mathcal{G}$ is reproduced in Figure 4.

This work only considers modifications to the stochastic storage stage of the algorithm and uses identical algorithms for the customer allocation and routing aspects of the problem. Hence we only review the stochastic storage model presented in [11] and omit the description of the other algorithms.

Stochastic Storage Model (SSM) The first stage captures the cost and demand objectives precisely but approximates the routing aspects. In particular, the model only considers the time to move the commodity from the repository to a customer, not the maximum delivery times. Let D be a set of delivery triples of the form $\langle source, destination, quantity \rangle$. The delivery-time component of the objective is replaced by

$$W_y * \sum_{\langle s,d,q \rangle \in D} T_{s,d} * q/VC$$

Figure 5 presents the SSM formulation, which scales well with the number of disaster scenarios because the number of integer variables only depends on the number of sites n. The meaning of the decision variables is explained in the figure. Once the storage and customer allocation are computed, the uncertainty is

Variables:

$Stored_{i \in 1..n} \in (0, RC_i)$ - Units stored
$Open_{i \in 1..n} \in \{0, 1\}$ - More than zero units stored flag

$StoredSaved_{s \in 1..a, i \in 1..n} \in (0, C_{s,i})$ - Units used at the storage location
$StoredSent_{s \in 1..a, i \in 1..n} \in (0, RC_i)$ - Total units shipped to other locations
$Incoming_{s \in 1..a, i \in 1..n} \in (0, C_{s,i})$ - Total units coming from other locations
$Unsatisfied_{s \in 1..a, i \in 1..n} \in (0, C_{s,i})$ - Demand not satisfied
$Sent_{s \in 1..a, i \in 1..n, j \in 1..n} \in (0, RC_i/VC)$ - Trips needed from i to j

Minimize:

$$W_x * \sum_{s \in 1..a} P_s * \sum_{i \in 1..n} Unsatisfied_{s,i} +$$
$$W_y * \sum_{s \in 1..a} P_s * \sum_{i \in 1..n} \sum_{j \in 1..n} T_{s,i,j} * Sent_{s,i,j} +$$
$$W_z * \sum_{i \in 1..n} (RI_i * Open_i + RM_i * Stored_i)$$

Subject To:

$$\sum_{i \in 1..n} (RI_i * Open_i + RM_i * Stored_i) \leq B$$
$$RC_i * Open_i \geq Stored_i \quad \forall i$$
$$StoredSaved_{s,i} + Incoming_{s,i} + Unsatisfied_{s,i} = C_{s,i} \quad \forall s, i$$
$$StoredSaved_{s,i} + StoredSent_{s,i} \leq Stored_i \quad \forall s, i$$
$$\sum_{j \in 1..n} VC * Sent_{s,i,j} = StoredSent_i \quad \forall s, i$$
$$\sum_{j \in 1..n} VC * Sent_{s,j,i} = Incoming_i \quad \forall s, i$$
$$StoredSaved_{s,i} + StoredSent_{s,i} = 0 \quad \forall s, i \text{ where } i \text{ not in } AR_s$$

Fig. 5. Stochastic Storage Model MIP Formulation

revealed and the second stage reduces to a deterministic multi-depot, multiple-vehicle capacitated routing problem whose objective consists in minimizing the latest delivery. One of the difficulties in this setting is that the customer demand is typically much larger than the vehicle capacity. As a result, the routing is solved in two steps. First each repository is considered independently and a number of vehicle trips are determined to serve the repositories customers (Repository Routing). A trip is a tour that starts at the depot, visits customers, returns to the depot, and satisfies the vehicle capacity constraints. Second how to route the vehicles to perform all the trips and minimize the latest delivery time is determined (Fleet Routing).

5 Modeling and Algorithmic Enhancements

The runtime results presented in [11] indicate that the Fleet Routing stage of the algorithm is the dominant factor in the algorithm runtime. However, for instances with more than 100 storage locations, the SSM quickly dominates the

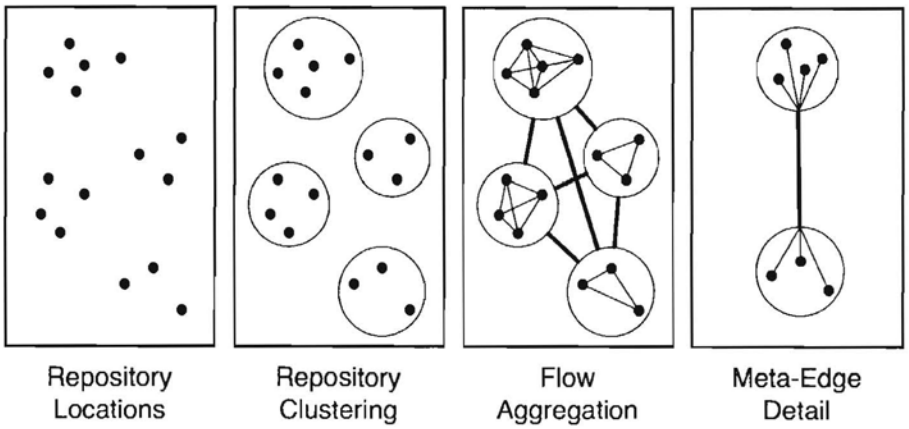

Fig. 6. Storage Clustering and Flow Aggregation.

runtime (see Section 6 for numerical evidence). This is particularly problematic for performance because the stochastic storage stage is the only algorithmic part that cannot be easily parallelized. In this section, we present two alternative models for the stochastic storage problem that provide significant benefits for scalability. Both stochastic storage models rely on a key observation: In the baseline algorithm (Figure 4), a customer allocation is computed in the SSM and then recomputed in the customer allocation stage. This means when a customer allocation stage is used only the storage decisions are a necessary output of the SSM. Both of these models achieve faster performance by approximating or ignoring the customer allocation in the stochastic storage problem. [3]

Aggregate Stochastic Storage Model (ASSM) In the SSM, the number of variables required for the customer allocation is quadratic in the number of repositories and multiplicative in the number of scenarios (i.e., an^2). The number of variables can easily be over one million when the number of repositories exceeds two hundred. Problems of this size can take up to 30 seconds to solve with a linear-programming solver and the resulting MIP can take several hours to complete. Our goal is thus to reduce the number of variables in the MIP solver significantly, without degrading the quality of the solutions too much.

The ASSM is inspired by observations of the solutions produced by the baseline algorithm. Customers are generally served by storage locations that are *nearby* and commodities are only transported over large distances in extreme circumstances. We exploit this observation by using a geographic clustering of the repositories. Given a mapping of the repositories into clusters, we say that

[3] It is also worth noting these model formulations assume that the transportation network is fully connected, however it is easy to imagine how these models could be extended to a disconnected network by reasoning on the connected components of the network.

two repositories are *nearby* if they are in the same cluster; otherwise the repositories are *far away*. Repositories that are nearby have a tightly-coupled supply and demand relationship so the model needs as much flexibility as possible in mapping the supplies to the demands. This flexibility is achieved by allowing commodities to flow between each pair of repositories within a cluster (as was done in SSM). When repositories are far away, the precise supply and demand relationship is not as crucial since the warehouse to customer relationship is calculated in the customer allocation stage of the algorithm. As a result, it is sufficient to reason about the aggregate flow moving between two clusters at this stage of the algorithm. The aggregate flows are modeled by introducing *meta-edges* between each pair of clusters. If some demands from cluster C_a must be met by storage locations from cluster C_b, then the sending repositories in C_b pool their commodities in a single *meta-edge* that flows from C_b to C_a. The receiving repositories in C_a then divide up the pooled commodities in the *meta-edge* from C_b to meet all of their demands. Additionally, if each *meta-edge* is assigned a travel cost, the *meta-edge* can approximate the number of trips required between two clusters by simply dividing the total amount of commodities by the vehicle capacity, as is the case for all the other flow edges. Figure 6 visually indicates how a clustering is used to generate the flow decision variables and how commodities can flow on *meta-edges* between customers in different clusters.

As stated above, the number of variables in the SSM is quadratic in the number of repositories. Given a clustering $Cluster_{i \in 1..c}$, the number of variables in the clustered storage model is (1) quadratic within each cluster (i.e., $\sum_{i=1}^{c} Cluster_i^2$); (2) quadratic in the number of clusters, (i.e., c^2); (3) and linear in the repositories connections to the clusters (i.e., $2nc$). The exact number of variables clearly depends on the considered clustering. However, given a specific number c of clusters, a lower bound on the number of variables is obtained by dividing the repositories evenly among the all clusters, and the best possible variable reduction on a problem of size n with c clusters is thus $a(\frac{n^2}{c} + 2nc + c^2)$.

Given a clustering $Cluster_{i \in 1..c}$ and cluster to cluster travel times $CT_{s,c,c}$, the Clustered Storage MIP is presented in Figure 7. The meaning of the decision variables is explained in the figure.

Sequential Stochastic Storage Model (SSSM) The ASSM significantly decreases the number of variables but it still requires creating a quadratic number of variables for the repositories inside each cluster. Since this is multiplied by the number of scenarios, the resulting number of variables can still be prohibitive for very-large instances.

We now investigate another approach whose idea is to decompose the objective function. It relies on the observation that, in practice, policy makers often set the values of W_x, W_y, W_z such that the objective is lexicographic in demand, delivery times, and money (i.e., $W_x \gg W_y \gg W_z$). Let us contemplate what this means for the behavior of the model algorithm as the budget parameter B is varied. With a lexicographic objective, the model will first try to meet as many demands as possible. If the demands can be met, it will reduce delivery times until it cannot be reduced further or the budget is exhausted. As a re-

Calculate:

$CS_c = \sum_{i \in Cluster_c} RC_i$ - Total storage in cluster c

$CD_{s,c} = \sum_{i \in Cluster_c} C_{s,i}$ - Total demand in cluster c

Variables:

$Stored_{i \in 1..n} \in (0, RC_i)$ - Units stored

$Open_{i \in 1..n} \in \{0, 1\}$ - More than zero units stored flag

$Unsatisfied_{s \in 1..a, i \in 1..n} \in (0, C_{s,i})$ - Demand not satisfied

$Incoming_{s \in 1..a, i \in 1..n, k \in 1..c} \in (0, C_{s,i})$ - Units received from cluster k to repository i

$Outgoing_{s \in 1..a, i \in 1..n, k \in 1..c} \in (0, RC_i)$ - Units sent from repository i to cluster k

$Sent_{s \in 1..a, i \in 1..n, j \in 1..n} \in (0, min(RC_i, C_{s,j}))$ - Units sent from repository i to repository j

$Link_{s \in 1..a, i \in 1..c, j \in 1..c} \in (0, min(CS_i, CD_{s,j}))$ - Units sent from cluster i to cluster j

Minimize:

$$W_x * \sum_{s \in 1..a} P_s * \sum_{i \in 1..n} Unsatisfied_{s,i} +$$

$$W_y * \sum_{s \in 1..a} P_s * \sum_{k \in 1..c} \sum_{i \in Cluster_i} \sum_{j \in Cluster_i} T_{s,i,j} * Sent_{s,i,j}/VC +$$

$$W_y * \sum_{s \in 1..a} P_s * \sum_{i \in 1..c} \sum_{j \in 1..c} CT_{s,i,j} * Link_{s,i,j}/VC +$$

$$W_z * \sum_{i \in 1..n} (RI_i * Open_i + RM_i * Stored_i)$$

Subject To:

$$\sum_{i \in 1..n} (RI_i * Open_i + RM_i * Stored_i) \leq B$$

$$RC_i * Open_i \geq Stored_i \quad \forall i$$

$$\sum_{j \in 1..n} Sent_{s,j,i} + \sum_{j \in 1..c} Incoming_{s,i,c} + Unsatisfied_{s,i} = C_{s,i} \quad \forall s, i$$

$$\sum_{j \in 1..n} Sent_{s,i,j} + \sum_{j \in 1..c} Outgoing_{s,i,c} \leq Stored_i \quad \forall s, i$$

$$\sum_{r \in Cluster_i} Outgoing_{s,r,j} = Link_{s,i,j} \quad \forall s, i, j$$

$$\sum_{r \in Cluster_j} Incoming_{s,r,i} = Link_{s,i,j} \quad \forall s, i, j$$

$$\sum_{r \in Cluster_i} Outgoing_{s,r,j} = \sum_{r \in Cluster_j} Incoming_{s,r,i} \quad \forall s, i, j$$

$Sent_{s,i,j} = 0 \quad \forall s, i, k$ where i not in AR_s

$Outgoing_{s,c,i} = 0 \quad \forall s, i, c$ where i not in AR_s

Fig. 7. Aggregate Stochastic Storage Model MIP Formulation

Calculate:

$SD_s = \sum_{i \in 1..n} C_{s,i}$ - Total demand

Variables:

$Stored_{i \in 1..n} \in (0, RC_i)$ - Units stored
$Open_{i \in 1..n} \in \{0, 1\}$ - More than zero units stored flag
$Used_{s \in 1..a} \in (0, SD_s)$ - Units used

Minimize:

$$\sum_{s \in 1..a} P_s * (SD_s - Used_s)$$

Subject To:

$$RC_i * Open_i \geq Stored_i \quad \forall i$$
$$\sum_{i \in AR_s} Stored_i \geq Used_s \quad \forall s$$
$$\sum_{i \in 1..n} (RI_i * Open_i + RM_i * Stored_i) \leq B$$

Fig. 8. Sequential Stochastic Storage Model MIP Formulation

sult, the optimization with a lexicographic objective exhibits three phases as B increases. In the first phase, the satisfied demands, routing times, and costs increase steadily. In the second phase, the satisfied demands remain at a maximum, the routing times decrease, and the costs increase. In the last phase, the satisfied demands remain at a maximum, the routing times remain at a minimum, and the costs plateau even when B increases further. The experimental results from [11] confirm this behavior.

The SSSM assumes that the objective is lexicographic and solves the first phase with a much simpler (and faster) model. The goal of this phase is to use the available budget in order to meet the demands as best possible and it is solved with a two-stage stochastic allocation model that ignores the customer allocation and delivery time decisions. Since each scenario s has a total demand SD_s that must be met, we simply need to maximize the expected amount of demands that can be met, conditioned on the stochastic destruction of storage locations. Figure 8 presents such a model. The meaning of the decision variables is explained in the figure.

During the first phase, the model in Figure 8 is identical to the SSM with a lexicographic configuration. But the model does not address the delivery times at all, since this would create a prohibitive number of variables. To compensate for this limitation, we use another model whose idea can be summarized by the following greedy heuristic: "if all the demands can be met, use the remaining budget to store as much additional commodity as possible". This greedy heuristic is calculated by cobmining the SSSM with an additional MIP model SSSM-B, which is presented in Figure 9. SSSM-B utilizes the remaining budget while enforcing the decisions of the first step by setting the lower bound of the

Variables:

$StoredEx_{i \in 1..n} \in (Stored_i, RC_i)$ - Units stored

$OpenEx_{i \in 1..n} \in \{0, 1\}$ - More than zero units stored flag

Maximize:

$\sum_{i \in 1..n} StoredEx_i$

Subject To:

$$RC_i * OpenEx_i \geq StoredEx_i \quad \forall i$$

$$\sum_{i \in 1..n} (RI_i * OpenEx_i + RM_i * StoredEx_i) \leq B$$

Fig. 9. Sequential Stochastic Storage Model MIP Formulation - B

$StoredEx_i$ variables to the value of the $Stored_i$ variables, which were computed by the SSSM. This approximation is rather crude but produces good results on actual instances (see Figures 10 & 11 in Section 6). Our future work will investigate how to improve this by taking account of customer locations, while still ignoring travel distances.

The resulting approach is much less flexible than the SSM and ASSM approaches because it ignores the weighting factors W_x, W_y, W_z. However, it produces a significant increase in performance by decreasing the number of decision variables asymptotically (i.e., from quadratic to linear). The asymptotic reduction is essential for scaling the algorithm to very large instances.

Note that it is well-known in the goal programming community that lexographic multi-objective programs can be solved by a series of single-objectice problems [12]). The sub-objectives are considered in descending importance and, at each step, one sub-objective is optimized in isolation and side constraints are added to enforce the optimization of the previous steps. Our decomposed storage model follows the same schema, except that the second step is necessarily approximated due to its size.

6 Benchmarks & Results

Benchmarks The benchmarks were produced by Los Alamos National Laboratory and are based on the infrastructure of the United States. The disaster scenarios were generated by state-of-the-art hurricane simulation tools similar to those used by the National Hurricane Center. Their sizes are presented in Table 1 (The table also depicts the algorithm parameters). The *Trip Lower Bound* is simply the total amount of commodities that are moved between locations divided by the vehicle capacity. This value is included because it is a good metric for the routing difficulty of a benchmark. The amount of commodities that need to be moved can vary significantly from scenario to scenario; Therefore, we present both the smallest and the largest trip bounds across all the scenarios. Benchmarks 3 and 6 feature scenarios where the hurricane misses the region; this results in the minimum trip bound being zero. This is important since any

Benchmark	n	m	a	Min Trip Lower Bound	Max Trip Lower Bound	CA Timeout	LNS Timeout	Clusters
BM1	25	4	3	6	27	30	10	4
BM2	25	5	3	60	84	30	20	4
BM3	25	5	3	0	109	30	20	4
BM4	30	5	3	35	109	30	20	4
BM5	100	20	3	82	223	90	200	4
BM6	25	5	18	0	140	30	20	4
BM7	30	10	18	7	23	30	20	4
BM9	250	10	18	7	23	250	45	10
BM10	500	20	18	13	45	-	180	-
BM12	1000	20	3	64	167	-	300	-

Table 1. SCAP Benchmark Statistics

algorithm must be robust with respect to empty disaster scenarios which arise in practice when hurricanes turn away from shore or weaken prior to landfall. All of the experimental results have fixed values of W_x, W_y, and W_z satisfying the field constraint $W_x \gg W_y \gg W_z$ and we vary the value of the budget B to evaluate the algorithm (as was done in [11]). The results are consistent across multiple weight configurations, although there are variations in the problem difficulties. It is also important to emphasize that, on these benchmarks, the number of trips in the worst case are larger than the *Max Trip Lower Bound* and thus produce routing problems of significant sizes.

The Algorithm Implementation and the Baseline Algorithm The algorithms were implemented in the COMET system [14] and the experiments were run on Intel Xeon CPU 2.80GHz machines running 64-bit Linux Debian. To validate our results, we compare our proposed storage models with those of the previous study [11]. Our routing time results also include the solution from the greedy agent-based algorithm proposed in [11] which mimics how actual decision makers operate in the field and thus provides a sense of improvement and scale in solution quality. The agent-based algorithm uses the storage model but builds a routing solution without any optimization. Each vehicle works independently to deliver as much commodity as possible.

Baseline Efficiency Results Table 2 depicts the runtime results for the baseline algorithm proposed in [11] . In particular, the table reports, in average, the total time in seconds for all scenarios (T_1), the total time when the scenarios are run in parallel (T_∞), the time for the storage model (STO), the customer allocation model (CA), the repository routing (RR), the aggregate fleet routing (AFR), and fleet routing (FR). The first three fields(T_1, T_∞, STO) are averaged over ten identical runs on each of the budget parameters. The last four fields (CA, RR, AFR, FR) are averaged over ten identical runs on each of the budget parameters and each scenario. Since these are averages, the times of the individual components do not sum to the total time. The results show that the approach

Benchmark	$\mu(T_1)$	$\sigma(T_1)$	$\mu(T_\infty)$	$\sigma(T_\infty)$	$\mu(STO)$	$\sigma(STO)$	$\mu(CA)$	$\mu(RR)$	$\mu(AFR)$	$\mu(FR)$
BM1	89.89	21.90	39.96	13.01	**0.9293**	**0.4670**	9.257	0.1746	10.057	10.13
BM2	169.1	35.93	66.02	10.47	**0.5931**	**0.2832**	16.67	0.1956	19.26	20.00
BM3	98.58	14.51	61.07	13.79	**0.3557**	**0.1748**	7.225	0.1050	12.04	13.33
BM4	184.2	26.25	68.76	5.163	**0.8892**	**0.3940**	21.24	0.2075	19.58	20.00
BM5	1308	62.01	520.5	32.70	**46.70**	**21.31**	90.87	1.225	128.0	200.0
BM6	723.5	58.76	75.34	3.079	**5.165**	**3.076**	10.81	0.1281	13.35	15.56
BM7	832.0	97.05	75.13	13.31	**16.15**	**5.153**	5.500	0.4509	19.31	20.00
BM9	15972	14245	13114	14205	**12926**	**14205**	79.93	1.426	41.29	44.96
BM10	-	-	-	-	-	-	-	-	-	-
BM12	-	-	-	-	-	-	-	-	-	-

Table 2. SCAP Benchmark Runtime Statistics (Seconds)

Benchmark	$\mu(T_1)$	$\sigma(T_1)$	$\mu(T_\infty)$	$\sigma(T_\infty)$	$\mu(STO)$	$\sigma(STO)$	$\mu(CA)$	$\mu(RR)$	$\mu(AFR)$	$\mu(FR)$
BM1	89.77	22.19	39.25	13.28	**0.5464**	**0.2389**	9.043	0.1791	10.04	10.12
BM2	169.4	35.91	65.93	10.49	**0.4846**	**0.1850**	16.81	0.2084	19.26	20.00
BM3	98.73	14.49	61.15	13.81	**0.3986**	**0.1609**	7.245	0.1092	12.05	13.33
BM4	182.8	24.28	69.74	3.822	**0.6950**	**0.3717**	20.88	0.2122	19.56	20.00
BM5	1266	70.41	487.4	35.18	**18.40**	**7.700**	90.88	0.8691	123.9	200.0
BM6	714.86	59.04	73.28	1.032	**3.130**	**1.041**	10.57	0.09642	13.27	15.56
BM7	823.6	98.79	67.95	12.99	**8.849**	**2.666**	5.479	0.4475	19.28	20.00
BM9	5534	974.7	1226	567.8	**878.4**	**567.8**	169.9	0.9652	41.50	45.01
BM10	-	-	-	-	-	-	-	-	-	-
BM12	-	-	-	-	-	-	-	-	-	-

Table 3. Clustered Storage Runtime Statistics (Seconds)

scales well with problems with 100 repositories or less. However, benchmark 9 (250 repositories) clearly indicates that the runtime of the storage model has exploded and become the dominating factor of the algorithm. Benchmarks 10 and 12 are unsolvable due to memory issues (these models require over 3,000,000 variables).

ASSM Quality & Efficiency Results Table 3 depicts the improvement of our ASSM for the SCAP algorithm. Observe the 50% reduction in runtime of the storage model (STO) and uniform benefits of our approach which systematically delivers that reduction on the larger benchmarks. In this study we cluster the repositories using their geographic locations using ten samples of the k-means algorithm. The sample with the smallest mean sum is used in the clustered storage model. The distances between clusters are calculated on a scenario-by-scenario basis using the average distance between all pairs of points in each cluster.

The runtime benefits of the clustering algorithm are largely due to the reduction in the number of variables in the model. Section 5 analyzed the variable

Benchmark	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM9	BM10	BM12
SSM	1875	1875	1875	2700	30000	11250	16200	1125000	-	-
ASSM	1116	1206	1248	1470	12246	7344	9576	237420	-	-
Lower Bound	1101	1101	1101	1443	9948	6606	8658	204300	-	-

Table 4. Clustered Problem size compared to the Baseline Algorithm

Benchmark	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM9	BM10	BM12
Relative Change(%)	-0.356	0.0834	-0.108	-0.504	-0.887	3.54	-0.308	1.16	-	-
Greedy Change(%)	56.4	43.1	73.7	52.0	64.0	92.2	55.5	200	-	-

Table 5. Clustered Quality compared to the Baseline Algorithm

reduction and pointed out that the reduction is tightly coupled with the clustering. Due to geographic considerations in these instances, the clustering exhibits great variation from instance to instance and it is important to report the actual reduction in problem size. Table 4 presents the number of variables of the SSM and the ASSM, as well as the lower bound on the number of variables. Observe that the benefits become more significant as the problem size grows and the runtime results confirm this.

Table 5 describes the relative changes in routing times from the SSM. The quality degradation of the greedy algorithm is also presented to provide a sense of scale. Because the ASSM is a courser approximation of the travel time, some decrease in routing quality is expected. It is impressive that the reduction in quality is not significant (especially when compared with the greedy algorithm).

It is also surprising that sometimes the clustering model improves the quality of the routing solution. This is a result of the fact that the travel time objective is only approximated in *all* of the stochastic storage models. When there are large distances between nodes, the ASSM's meta-edges provide a more accurate estimate of the number of trips needed between two clusters.

Unfortunately this model still suffers from the same memory issues as the SSM and is unable to solve Benchmarks 10 and 12. Figure 10 visually summarizes the time and quality tradeoff of the ASSM and the SSM.

SSSM Quality & Efficiency Results Table 6 depicts the improvement of our SSSM for the SCAP algorithm. Observe the consistent reduction in runtime of the storage model (STO), which runs about 1000 times faster than the SSM on BM9.

Table 7 describes the relative change in routing times from the SSM. The quality degradation of the greedy algorithm is also presented to provide a sense of scale. Because the SSSM has no information about the travel time, some decrease in routing quality is expected. Again, it is impressive that the reduction is so small (especially when compared with the greedy algorithm). Note that some policy makers may be concerned by the 6.6% increase in delivery time in benchmark 9 and may prefer to use the SSM. However, some types of disasters

Benchmark	$\mu(T_1)$	$\sigma(T_1)$	$\mu(T_\infty)$	$\sigma(T_\infty)$	$\mu(STO)$	$\sigma(STO)$	$\mu(CA)$	$\mu(RR)$	$\mu(AFR)$	$\mu(FR)$
BM1	94.97	24.38	41.32	12.79	**0.1166**	**0.06717**	11.46	0.1943	9.819	10.00
BM2	169.3	36.04	65.59	10.43	**0.1624**	**0.07953**	16.86	0.2375	19.25	20.00
BM3	98.85	14.31	60.97	13.76	**0.1705**	**0.09249**	7.280	0.1260	12.12	13.34
BM4	183.8	24.26	69.15	3.527	**0.2514**	**0.1918**	21.27	0.2547	19.59	20.00
BM5	1240	69.07	468.6	33.70	**2.125**	**0.8614**	90.94	0.8141	120.6	200.0
BM6	719.8	56.91	70.46	0.08338	**0.3516**	**0.08734**	11.06	0.08603	13.23	15.56
BM7	810.5	100.8	59.54	13.51	**0.7089**	**0.1497**	5.397	0.3703	19.17	20.00
BM9	5963	439.0	366.5	7.134	**13.19**	**4.661**	239.6	2.166	42.58	45.00
BM10	32048	1708	1921	108.9	**17.34**	**22.02**	1385*	8.659	175.7	180.0
BM12	18201	73.11	6146	46.54	**14.12**	**0.2223**	5485*	14.28	227.3	300.0

Table 6. Decomposed Storage Runtime Statistics (Seconds)

Benchmark	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM9	BM10	BM12
Relative Change(%)	3.61	-0.0549	-0.0371	0.524	0.524	6.22	-0.527	6.64	-	-
Greedy Change(%)	60.6	43.5	74.5	52.9	67.2	103	55.5	227	78.5	91.5

Table 7. Decomposed Quality compared to the Baseline Algorithm

require immediate response where every minute is valuable. In those extreme situations, the decomposed storage model provides a much faster alternative to the baseline algorithm. These new algorithms allow the policy maker to choose on a case-by-case basis which is preferable, a more immediate response or a higher quality solution.

The lack of information about travel time is an advantage for the memory usage of the SSSM. Only three pieces of the problem specification need to be considered, the repository information, scenario demands, and scenario damage. This resolves the memory issues faced by the other models by loading the scenario travel time separately for each scenario. This allows the SSSM to scale to the largest benchmarks. Figure 10 visually summarizes the runtime and quality tradeoff of the SSSM and the SSM.

Due to the enormous size of benchmarks 10 and 12, the customer allocation stage of the algorithm does not return a feasible solution within 1000 seconds. To resolve this difficulty, we simply ignore the integer variables and solve the linear programming relaxation of the same problem, which is then rounded. As Table 6 indicates just solving a linear programming relaxation of these problems can take over ten minutes. Additionally, to make the runtime of the SSSM stable on the largest instnaces the solver is terminated whenever the optimiality gap is reduced to 0.05%.

Behavioral Analysis of SSSM The SSSM ignores the algorithm parameters W_x, W_y, W_z and implicitly assumes the field constraint $W_x \gg W_y \gg W_z$. Although the other storage models are more flexible in this regard, for the purpose of this study all the storage models are configured for this field constraint. This means that the

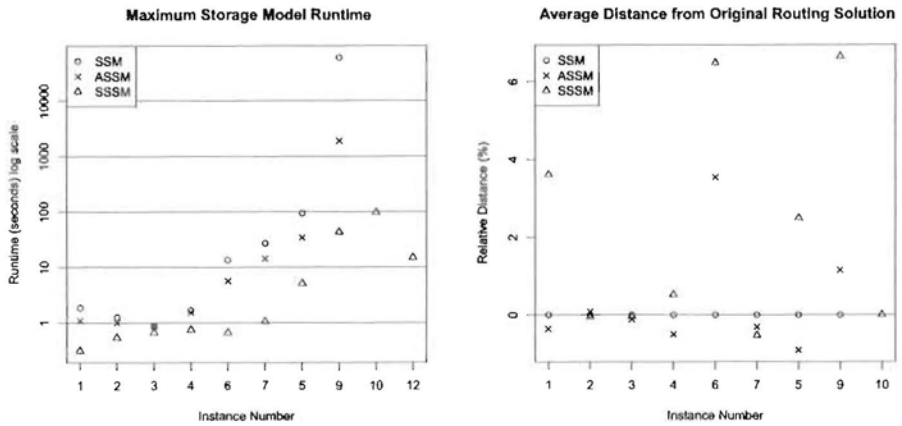

Fig. 10. Runtime and Quality Tradeoffs

storage decisions for the SSSM will be exactly the same as the SSM until all of the demands are met. Once all of the demands are satisfied, the SSSM will degrade because it cannot determine how to use additional funds to decrease the delivery time. However, as the budget increases it will approach the same solution as the SSM because these solutions correspond to storing commodities at all of the repositories. Figure 11 presents the experimental results on benchmark 6 which exhibits this behavior most dramatically (other benchmarks are less pronounced and omitted for space reasons). The graph on the left shows how the satisfied demand increases with the budget while the graph on the right shows how the last delivery time changes. We can see that as the satisfied demand increase the routing time of both algorithms is identical until the total demand is met. At that point, the routing times diverge as the travel distance becomes an important factor in the objective and re-converge as the budget approaches its maximum and all of the respositories are storing commodities. These results confirm our behavioral expectation. The experimental results also demonstrate that the degradation of the decomposed model is not significant when compared to the choices made by the greedy routing algorithm.

7 Conclusion

This paper studied the scalability of a problem in the field of humanitarian logistics, the Single Commodity Allocation Problem (SCAP). The SCAP models the strategic planning process for disaster recovery with stochastic last mile distribution. The paper proposed two new stochastic storage models that produce high quality solutions to real-world benchmarks that until this work were unsolvable. The algorithms use spacial and objective decompositions to exploit the problem structure and speedup stochastic storage decisions. The experimental

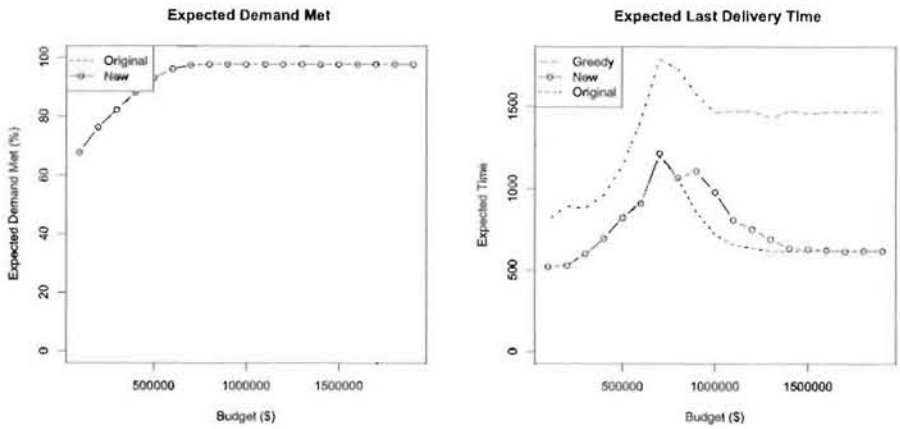

Fig. 11. Varying the Budget on Benchmark 6 with Decomposition

results on water allocation benchmarks indicate that the algorithm is: (1) practical from a computational standpoint; (2) produce significant scalability over previous work; (3) delivers better performance than existing relief delivery procedures. This work is currently deployed at Los Alamos National Laboratory as part of the National Infrastructure Simulation and Analysis Center (NISAC). It is being used to aid federal organizations such as the Department of Energy and the Department of Homeland Security in preparing for and responding to disasters.

References

1. B. Balcik, B. Beamon, and K. Smilowitz. Last mile distribution in humanitarian relief. *Journal of Intelligent Transportation Systems*, 12(2):51–63, 2008.
2. Glay Barbarosoglu, Linet zdamar, and Ahmet evik. An interactive approach for hierarchical analysis of helicopter logistics in disaster relief operations. *European Journal of Operational Research*, 140(1):118 – 133, 2002.
3. B. Beamon. Humanitarian relief chains: Issues and challenges. *34th International Conference on Computers & Industrial Engineering*, pages 77–82, 2008.
4. L. Bianchi, M. Dorigo, L.Gambardella, and W. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8(2), 2009.
5. Ann Melissa Campbell, Dieter Vandenbussche, and William Hermann. Routing for relief efforts. *Transportation Science*, 42(2):127–145, 2008.
6. Serhan Duran, Marco Gutierrez, and Pinar Keskinocak. Pre-positioning of emergency items worldwide for care international. *submitted to Interfaces*, 2008.
7. Fritz institute. http://www.fritzinstitute.org, 2008.
8. United States Government. The federal response to hurricane katrina: Lessons learned, 2006.
9. P. Griffin, C. Scherrer, and J. Swann. Optimization of community health center locations and service offerings with statistical need estimation. *IIE Transactions*, 2008.

10. D. Gunnec and F. Salman. A two-stage multi-criteria stochastic programming model for location of emergency response and distribution centers. In *INOC*, 2007.
11. Pascal Van Hentenryck, Russell Bent, and Carleton Coffrin. Strategic planning for disaster recovery with stochastic last mile distribution. In Andrea Lodi, Michela Milano, and Paolo Toth, editors, *CPAIOR*, volume 6140 of *Lecture Notes in Computer Science*, pages 318–333. Springer, 2010.
12. James P. Ignizio. A review of goal programming: A tool for multiobjective analysis. *The Journal of the Operational Research Society*, 29(11):pp. 1109–1119, 1978.
13. Peter Kall and Stein W. Wallace. *Stochastic Programming (Wiley Interscience Series in Systems and Optimization)*. John Wiley & Sons, 1995.
14. Comet 2.1 User Manual. Dynadec website. http://dynadec.com/.
15. Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, Pennsylvania, 2001.
16. L. Van Wassenhove. Humanitarian aid logistics: supply chain management in high gear. *Journal of the Operational Research Society*, 57(1):475–489, 2006.